EdTech Assignment Tracker – Part A: System Design

Design the system architecture and document:

- For frontend I used HTML, CSS, and JavaScript for interaction.
- Implemented using FastAPI, a high-performance web framework for backend.
- The database I used is SQLite.
- Authentication was done by using Role-based access control using JWT tokens.
- For Communication I used RESTful API for frontend-backend interaction.

Core Entities and Relationships (table descriptions):

Entity	Fields
User	id, name, email, password_hash, role (student/teacher)
Assignment	id, title, description, due_date, created_by (FK to User)
Submission	id, assignment_id (FK), student_id (FK), content, timestamp

Relationships:

- One Teacher → Many Assignments
- One Student → Many Submissions
- One Assignment → Many Submissions

API endpoints for the following actions:

Action	Method	Endpoint	Access
Signup/Login	POST	/auth/signup, /auth/login	All Users
Create Assignment	POST	/assignments/	Teacher
Submit Assignment	POST	/submissions/assignments/{id}/submit	Student
View Submissions	GET	/submissions/assignments/{id}	Teacher

Teacher creates an assignment:

Teacher will have different login and when we click on create assignment. It will ask Title, Description and Date when the assignment posted.

Student submits assignment:

When student login to the site it will automatically directed to the submit.html page. There the student can write the assignment and can submit.

Teacher views submissions:

Teacher can click the ID and view assignment, then assignments will be displayed.

Authentication Strategy:

The system uses JWT-based authentication. During login, a token is issued which contains user role and ID. This token is included in headers for each API request. Routes are protected using role-based logic:

- Teachers can create/view assignments and submissions.
- Students can submit assignments.

Scalability Suggestions

To scale the system effectively, it's recommended to switch from SQLite to PostgreSQL for better support of concurrent access. Background tasks like file uploads can be managed using Celery. Introducing Redis will help cache frequently accessed data and reduce database load.

Key Points:

- Replace SQLite with PostgreSQL
- Use Celery for background tasks
- Add Redis for caching
- Deploy with Docker and Kubernetes
- Enable email notifications